Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В Непомнящий
подпись инициалы, фамилия

«___» _____ 20___ г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Построение полносвязанных сетей на базе коммутаторов Ethernet
Тема

09.04.01 «Информатика и вычислительная техника»
код и наименование направления

09.04.01.05 «Сети ЭВМ и телекоммуникации»
код и наименование магистерской программы

Научный руководитель	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	Ф.А. Казаков инициалы, фамилия
Выпускник	_____ подпись, дата		М.С. Русин инициалы, фамилия
Рецензент	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	В.Н. Коршун инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	Ф.А. Казаков инициалы, фамилия

Красноярск 2020

Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В Непомнящий
подпись инициалы, фамилия

«__» _____ 20__ г.

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме магистерской диссертации

Студенту Русину Михаилу Сергеевичу.

Группа: КИ18-01-5м Направление (специальность): 09.04.01 Информатика и вычислительная техника.

Тема выпускной квалификационной работы: «Построение полносвязанных сетей на базе коммутаторов Ethernet».

Утверждена приказом по университету _____ № _____

Руководитель ВКР: Ф.А. Казаков, доцент, канд. техн. наук.

Исходные данные для ВКР: задание на магистерскую диссертацию.

Перечень разделов: обзор существующих аналогов, проектирование алгоритма протокола, разработка и тестирование протокола.

Перечень графических материалов: презентация, блок-схема алгоритма.

Руководитель ВКР _____ Ф.А. Казаков
подпись

Задание принял к исполнению _____ М.С. Русин
подпись

РЕФЕРАТ

Выпускная квалификационная работа по теме «Построение полносвязанных сетей на базе коммутаторов Ethernet» содержит 48 страниц текстового документа, 20 рисунков, 2 таблицы, 13 использованных источников.

КОММУТАЦИЯ, ЦЕНТР ОБРАБОТКИ ДАННЫХ, ETHERNET, FABRIC PATH, SPB, TRILL.

Цель работы: повышение производительности сетей второго уровня путем использования алгоритма построения полносвязанных сетей на базе коммутаторов Ethernet.

Задачи:

– выполнить анализ предметной области;

– разработать алгоритм построения полносвязанных сетей;

– реализовать и протестировать алгоритм построения полносвязанных сетей.

Во введении раскрывается актуальность и значимость работы, ставится цель и задачи.

В первой главе рассматриваются существующие решения, а также анализируются их достоинства и недостатки.

Во второй главе описывается разработка алгоритма построения полносвязанных сетей на основе обзора аналогов из первой главы.

Третья глава посвящена программной реализации и тестированию заявленных функций разработанного алгоритма.

В результате работы создан, реализован и протестирован алгоритм построения полносвязанных сетей на базе коммутаторов Ethernet.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время существует большая потребность в технологиях, которые могли бы позволить использовать все доступные каналы передачи данных в сетях второго уровня. Протоколы семейства STP, которые были созданы для повышения отказоустойчивости сетей, имеют ряд существенных недостатков. Среди них можно отметить блокировку портов на коммутаторах для перестроения топологии, что ведет к простаиванию сетей [1]. Также стоит отметить, что потеря служебных BPDU пакетов неминуемо приведет к выходу сети из строя. Для решения этих проблем были разработаны протоколы, которые совмещают в себе функции второго и третьего уровня сетей. Одними из представителей таких технологий являются SPB, TRILL и FabricPath. Они зарекомендовали себя надежной и функциональной заменой протоколу STP, но, как и любые другие алгоритмы, имеют свои недостатки.

Основной функцией, которая объединяет все вышеперечисленные протоколы, является возможность использования всех доступных каналов передачи данных внутри сетей, где они работают. Это позволяет равномерно распределять нагрузку и увеличивает скорость передачи данных. Но в то же время их объединяет один общий недостаток: размер заголовков, необходимых для работы протоколов, которые составляют от 40 до 60 байт, что является весьма больший значением в сравнении с стандартным заголовком Ethernet в 8 байт.

Анализ достоинств и недостатков SPB, TRILL и FabricPath позволяет разработать собственный алгоритм, который повторял бы главную идею этих технологий и устранял выявленные недостатки.

Актуальность выбранной темы обусловлена необходимостью создания полносвязанных сетей на основе Ethernet в связи с бурным развитием центров обработки данных и объемом передаваемого трафика.

Цель работы: разработка алгоритма построения полносвязанных сетей на базе коммутаторов Ethernet.

Для достижения указанной цели в работе решаются следующие **задачи**:

– выполнение аналитического обзора имеющихся аналогов;

– разработка алгоритма;

– реализация и тестирование протокола на основе ранее разработанного алгоритма.

Научная новизна: разработанный алгоритм позволяет использовать все доступные каналы передачи данных в сетях Ethernet.

Практическая ценность данного алгоритма заключается в том, что сети, построенные с использованием данного алгоритма, будут распределять нагрузку по всем доступным каналам, что должно улучшить работу сети в целом.

1 Обзор существующих аналогов

1.1 IEEE 802.1aq Shortest Path Bridging

Для обеспечения высокой надежности большинство корпоративных сетей и центров обработки данных должны обладать избыточными каналами связи между физическими узлами. Также это необходимо для динамического равномерного распределения ресурсов и свободного перемещения виртуальных машин (VM). Поскольку миграция VM должна проходить незаметно для пользователей, то и требования к сетям, в которых они работают весьма специфичны. Среди таких требований можно отметить следующие [2]:

– сеть не должна быть подвержена проблемам широковещательных штормов и сети подключенные к облаку не должны влиять на работу друг друга;

– технологии, работающие в сети, никак не должны мешать работе по передаче данных наиболее распространенных технологий, например IP;

– множественная адресация должна быть доступна, но необходим механизм борьбы с петлями;

– трафик внутри сети должен быть полностью оптимизирован для экономии полосы пропускания и должен балансировать нагрузку на каналы при избыточности связей;

– сеть центра обработки данных должна быть прозрачна для клиентов и нетребовательна к их топологии;

– сеть должна быть легко настраиваться и обслуживаться. Настройка клиентов должна проходить по принципу Plug-and-play.

Все эти функции выполняет протокол IEEE 802.1aq Shortest Path Bridging (SPB).

IEEE 802.1aq Shortest Path Bridging (SPB) обеспечивает многоканальную маршрутизацию в ячеистой сети Ethernet с использованием IS-IS в качестве протокола управления. Технология позволяет активировать все пути,

поддерживает пути с одинаковой метрикой и обеспечивает пересылку трафика по кратчайшим путям в ячеистой сети Ethernet. IEEE 802.1aq SPB обеспечивает более быструю конвергенцию, более высокую эффективность каналов и возможность строить большие сети второго уровня, чем обычные протоколы связующего дерева, такие как xSTP. SPB имеет следующие преимущества:

– быстрое развертывание технологии и легкое обслуживание;

– быстрое восстановление сети после сбоев;

– обычные коммутаторы могут подключаться к сети SPB через основные и резервные каналы;

– SPB поддерживает работу со старым оборудованием и предоставляет возможность плавного обновления сети. Помимо этого, технология совместима с множеством протоколов и стандартов.

Существует два варианта реализации SPB: SPBV и SPBM. SPBV считается первой версией протокола SPB и базировался на основе MSTP. Главной идеей является возможность строить большие сети, чего не позволяет MSTP. Это достигается путем построения маршрутов на основе VLAN. Для этого каждый канал между коммутаторами в сети SPBV определяется в отдельный VLAN и протокол состояния канала IS-IS изучает сеть на наличие коротких маршрутов. Однако более интересным и полезным, с точки зрения центров обработки данных, является версия SPBM.

SPBM использует в своей основе технологию 802.1ah-2008 Provider Backbone Bridges (PBB), т.е полную инкапсуляцию MAC-in-MAC. Перед описанием основной идеи и алгоритма стоит раскрыть несколько терминов [2].

Backbone link и access link. Канал, который соединяет магистральный базовый мост (Backbone Core Bridge, BCB) с магистральным краевым мостом (Backbone Edge Bridge, BEB), является магистральным (Backbone link). Канал, который соединяет BEB с сетью клиента, называется каналом доступа (Access link). После того, как кадры из сети клиента инкапсулированы в кадры по принципу MAC-in-MAC, они передаются по соответствующей магистральному

каналу BEB. После того, как кадры из сети SPBM декапсулированы, они передаются по соответствующей линии доступа BEB в соответствии с MAC-адресом клиента.

Backbone Core Bridge, BCB. BCB-коммутаторы являются базовыми узлами сети SPBM. BCB пересылают кадры MAC-in-MAC на основе B-MAC и B-VLAN. Они не изучают клиентские MAC-адреса (C-MAC). Это делает сеть SPBM легко масштабируемой.

Backbone Edge Bridge, BEB. BEB-коммутаторы являются граничными узлами сети SPBM. BEB инкапсулируют кадры клиента в кадры MAC-in-MAC, прежде чем перенаправить их в сеть SPBM. BEB также декапсулируют кадры MAC-in-MAC перед отправкой их на клиентские устройства.

B-MAC и B-VLAN - магистральные MAC-адреса (Backbone MAC addess, B-MAC) — это MAC-адреса мостов, связанных с другими мостами в SPBM сети. Магистральные VLAN (Backbone VLAN, B-VLAN) — это VLAN, назначенные поставщиком услуг (сетью SPBM) для передачи трафика клиента в сети SPBM. Для того чтобы кадры клиента передавались по сети SPBM, входной BEB инкапсулирует их в формате MAC-in-MAC. Во внешнем заголовке кадра MAC-адрес источника является B-MAC входного BEB, а MAC-адрес назначения является B-MAC выходного BEB. Все устройства в сети SPBM пересылают кадры MAC-in-MAC на основе целевого B-MAC и B-VLAN.

Сетевой порт клиента и сетевой порт провайдера. На BEB-коммутаторе сетевой порт клиента (Client network port, CNP) подключается к клиентам, а сетевой порт провайдера (Provider network port, PNP) подключается к сети SPBM.

Экземпляр службы Ethernet. Экземпляр службы Ethernet предоставляет услуги пересылки наборов пользовательских VLAN (C-VLAN). Чтобы передавать C-VLAN между клиентами через сеть SPBM, необходимо настроить экземпляры службы Ethernet на пограничных портах клиентов BEB и сопоставить их с VSI SPB.

LSDB. База данных состояний каналов (Link state database, LSDB) содержит информацию о состоянии всех каналов в сети SPBM.

PW. Дословно псевдопровода (Pseudo Wires). Это туннели MAC-in-MAC, установленные в сети SPBM для передачи клиентского трафика. PW устанавливаются между BEB-коммутаторами. BCB не устанавливают PW.

SPB VSI и I-SID. Экземпляр виртуального коммутатора SPB (SPB Virtual Switch Instance, SPB VSI) предоставляет сервис туннелирования MAC-in-MAC для экземпляров сервиса Ethernet. SPB VSI действует как виртуальный коммутатор. Он имеет все функции обычного коммутатора Ethernet, включая изучение MAC-адреса источника, устаревание MAC-адреса и переполнение. Каждый SPB VSI уникально идентифицируется по I-SID.

Сети SPBM состоят BEB и BCB. BEB соединяют клиентские сети с сетью SPBM. Клиентская сеть — это сеть второго уровня, которая имеет независимые сервисные функции. Такие обычно управляется и контролируется одной организацией и состоит из хостов и коммутационных устройств, которые никак не должны влиять и зависеть от сети SPBM. Базовая сеть состоит из коммутационных устройств с поддержкой ISIS-SPB и реализует межуровневое соединение второго уровня между сетями клиентов. На рисунке 1.1.1 показана типичная модель сети SPBM.

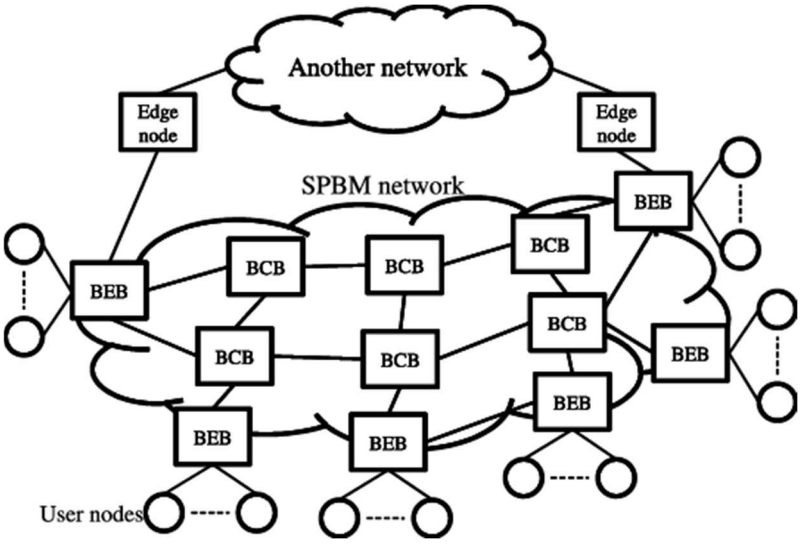

Рисунок 1.1.1 — Пример SPBM сети

SPBM имеет отдельные алгоритмы для одноадресного и многоадресного трафика. Под многоадресным трафиком понимается не только Broadcast-трафик, но и трафик с неизвестными Unicast-адресами и Multicast-адресами [3].

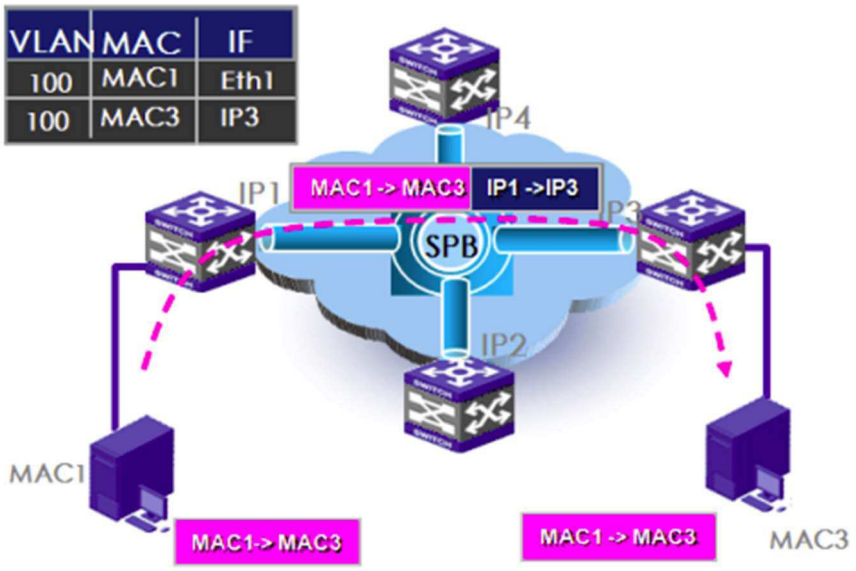

Рисунок 1.1.2 — Пример передачи одноадресного трафика

Для одноадресной пересылки трафика алгоритм выглядит следующим образом (рисунок 1.1.2):

– когда входной BEB получает кадр Ethernet, он выполняет следующие задачи: назначает кадр правильному SPB VSI на основе внешнего VLAN, узнает исходный MAC-адрес, ищет MAC-адреса назначения в таблице MAC-адресов SPB VSI;

– если исходящий интерфейс является портом транспортной сети, входной BEB инкапсулирует исходный кадр Ethernet в формате MAC-in-MAC. MAC-адрес источника во внешнем заголовке Ethernet является MAC-адресом входного BEB, MAC-адрес определения является MAC-адресом выходного BEB, а B-тег содержит B-VLAN VSI SPB;

– входной BEB отправляет кадр MAC-in-MAC в базовую сеть через сетевой порт провайдера;

– выходной BEB декапсулирует пакет и удаляет внешний заголовок Ethernet;

– выходной BEB ищет MAC-адрес назначения во внутреннем заголовке для локального исходящего интерфейса и отправляет кадр Ethernet из исходящего интерфейса на хост назначения.

Для пересылки многоадресного трафика SPBM имеет два метода:

– Head-end replication — реплицирует кадры на входном BEB для кадров, поступающих в сеть SPBM. Этот метод подходит для VSI SPB, которые имеют разреженный многоадресный трафик. Для это не требует, чтобы BCB поддерживали многоадресные записи FDB;

– Tandem replication — реплицирует кадры только на узле, где разветвляется дерево кратчайшего пути. Этот метод подходит для VSI SPB с плотным многоадресным трафиком. Это требует BCB для поддержки многоадресных записей FDB.

Как упоминалось выше SPBM использует инкапсуляцию. Для этого используется технология PBB. Итоговый кадр для передачи по сети SPBM выглядит согласно рисунку 1.1.3 [4].

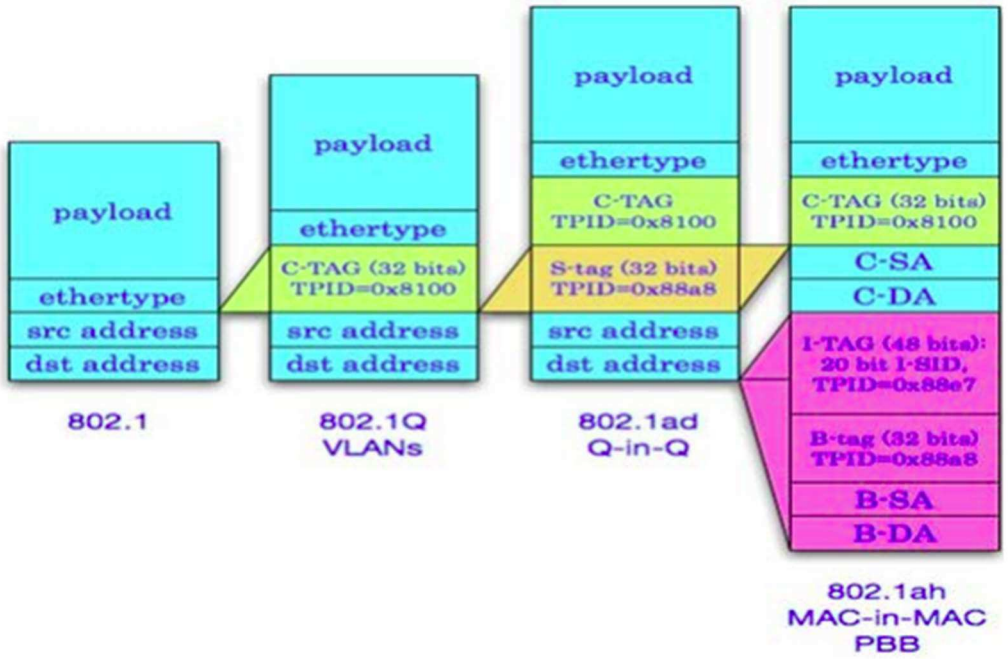

Рисунок 1.1.3 — Формат кадра протокола SPBM

Разработчика протокола SPBM удалось добиться главной цели. Благодаря их разработке удалось избавиться от ограничений второго уровня и протоколов семейства STP. Но среди достоинств есть и недостатки. Среди последних можно отметить использование инкапсуляции, ограниченный набор вендоров поддерживающих эту технологию (Extreme, Alcatel, HP) и использование служебного трафика для отслеживания состояния каналов и обмена информацией из таблиц LSDB.

1.2 TRILL

Открытый протокол TRILL (Transparent Interconnection of Lots of Links, IETF 6325) был разработан Инженерным советом интернета (IETF) для облегчения масштабирования систем и устранения всех сопутствующих проблем, чего невозможно добиться, используя протокол STP.

В качестве основы для управления трафиком протокола TRILL использует специальную версию протокола маршрутизации IS-IS и алгоритм Дейкстры (Shortest Path First, SPF) для построения коротких маршрутов [5].

В эпоху облачных вычислений центр обработки данных (ЦОД) обычно использует распределенную архитектуру для массового хранения данных, запросов и поиска. Поскольку технологии виртуализации широко используются в кластерных вычислениях, каждый сервер должен вычислять гораздо больше данных, чем раньше, и, следовательно, пропускная способность физического сервера увеличивается в несколько раз. Кроме того, виртуальные машины (VM) должны иметь возможность динамической миграции в пределах центра обработки данных, чтобы повысить надежность обслуживания, сократить расходы на эксплуатацию и обслуживание сети, а также обеспечить более гибкое развертывание IT-услуг.

Из-за этих нюансов традиционная иерархическая структура сети с доступом уровня 2 (xSTP) и с ядром уровня 3 (маршрутизация) не может

удовлетворить требования центров обработки данных. В настоящее время в ЦОД широко используется архитектура большого дерева второго уровня. TRILL помогает создать не блокирующую большую сеть уровня 2, которая поддерживает плавную миграцию виртуальных машин и может адаптироваться к растущим масштабам сети.

Разработчики TRILL выделяют следующие преимущества [6]:

– TRILL упрощает настройку и снижает затраты на управление. Для работы протокола требуется включить его на RB-коммутаторах и обозначить роли для портов (trunk подключаются к другим RB, а access к конечным устройствам). Так же в случае миграции серверов TRILL динамически адаптируется к изменениям, в то время как IP-сети должны быть перестроены;

– для отслеживания состояния каналов, быстрой сходимости и работы с большими сетями второго уровня TRILL использует специальную версию протокола IS-IS L2. Для уменьшения последствий циклов и BC-трафика протокол использует TTL в кадрах и RPF для удаления колец;

– TRILL обеспечивает высокую эффективность и производительности во время работы. Это достигается путем использования ECMP, пересылке трафика по кратчайшим маршрутам и ненадобностью изучать MAC-адреса конечных устройств коммутаторам уровня распределения.

Данный, и ему подобные протоколы, помогли достичь большого горизонтального трафика (east-west traffic). Это весьма актуально, ведь в подавляющем количестве сетевых архитектур используется распределенное хранение информации и виртуализация серверов, что в конечном итоге дает возможность для упрощенной миграции данных и серверов, снижая стоимость владения IT инфраструктуры [7].

Каждое устройство, входящее в сеть на основе технологии TRILL имеет специально обозначенную роль и соответствующие функции (рисунок 1.2.1).

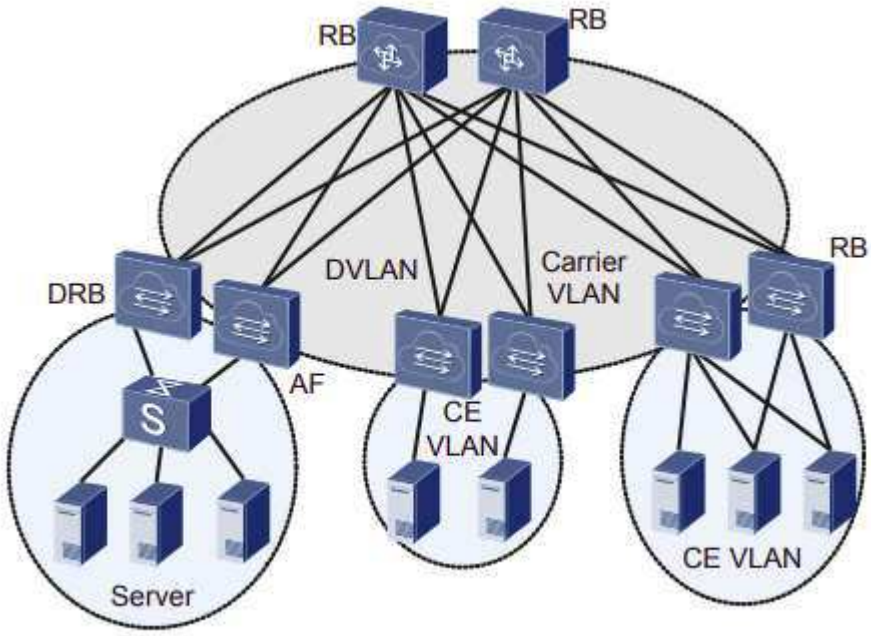

Рисунок 1.2.1 — Пример сети на базе TRILL с обозначением ролей

Такие роли можно разделить на три класса [6]:

1. Router Bridge (RB) — это коммутатор второго уровня, на котором работает TRILL. RB классифицируются на входной RB, транзитный RB и выходной RB в соответствии с их местоположением в сети TRILL. Входной RB указывает входной сигнал, с которого пакеты поступают в сеть TRILL. Транзитный RB указывает промежуточный узел, через который проходят пакеты в сети TRILL. Выходной RB указывает выходной пакет, из которого пакеты покидают сеть TRILL.

2. Designated Router Bridge (DRB) — это RB, который функционирует как транзитное устройство и выполняет специальные задачи в сетях TRILL. В широковещательной сети TRILL, если два RB находятся в одной и той же виртуальной локальной сети (VLAN), в качестве DRB выбирается RB, интерфейс которого с более высоким приоритетом DRB или большим MAC-адресом, когда они устанавливают отношения соседей. DRB связывается с каждым устройством в сети для синхронизации всех баз данных состояния канала (LSDB) в VLAN, освобождая каждые два устройства от связи для синхронизации LSDB. DRB

выполняет следующие функции: генерирует данные о состоянии каналов, обменивается блоками данных для синхронизации таблиц LSDB, выделяет отдельные VLAN для передачи пользовательского и служебного трафика, выбирает какому RB коммутатору работать в режиме AF.

3. Appointed Forwarder (AF) — это RB, выбранный DRB для пересылки пользовательского трафика. RB, который не является AF, не могут пересылать пользовательский трафик. Как показано на рисунке 1.2.1, петли могут возникать, если сервер подключен к сети TRILL двумя сетевыми адаптерами, но они не имеют функции распределения нагрузки. Следовательно, один из подключенных RB должен быть выбран для пересылки пользовательского трафика. Из всех подключенных к устройству RB коммутаторов в режиме AF может работать лишь один.

В описании DRB были упомянуто о существовании специальных VLAN. TRILL выделяет четыре вида VLAN с которыми он работает. Их описание и трафик, с которым они работают представлены в таблице 1.2.1 [6].

Таблица 1.2.1 — Таблица VLAN для протокола TRILL

Наименование VLAN	Описание	Трафик
VLAN классического Ethernet (CE)	VLAN предназначенный для подключения конечных устройств к сети TRILL	Native Ethernet пакеты
VLAN администрирования	Специальный CE VLAN для передачи служебного трафика в сети TRILL	Служебный трафик сети TRILL

Окончание таблицы 1.2.1

Наименование VLAN	Описание	Трафик
Несущий VLAN	Несущий VLAN передает контрольные пакеты TRILL и пакеты данных TRILL. На RB-коммутаторе может быть настроено максимум три несущих VLAN. Во входящем направлении CE пакеты инкапсулируются в пакеты TRILL в несущих VLAN. В исходящем направлении пакеты TRILL декапсулируются и восстанавливаются в CE пакеты.	Трафик с контрольными пакетами и данными.
Назначенный VLAN	Для объединения или разделения сетей TRILL могут быть настроены несколько несущих. Однако для пересылки пакетов управления и данных TRILL выбрана только одна VLAN-сеть. Выбранная VLAN называется назначенной VLAN.	Трафик с контрольными пакетами и данными.

Как было описано выше протокол TRILL использует инкапсуляцию-декапсуляцию для передачи трафика внутри сети. Обычный Ethernet кадр помещается внутрь пакета TRILL и имеет вид согласно рисунку 1.2.2 [8]. Такой подход имеет свои недостатки. Основным и наиболее важным является размер заголовка в 46 байт. Это означает, что из стандартных параметров MTU в 1500 байт приходится исключать в общей сложности 54 байта, из которых 46 байт-заголовки TRILL и 8 байт-заголовки Ethernet.

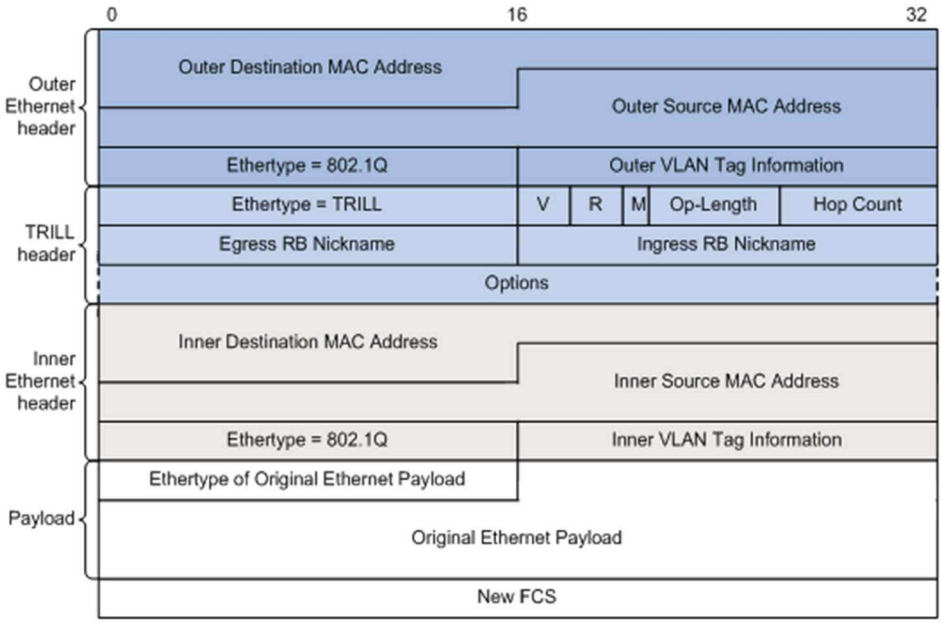

Рисунок 1.2.2 — Формат кадра протокола TRILL

Алгоритм работы протокола можно разобрать на небольшом примере. Получив кадр Ethernet, TRILL добавляет свой заголовок и внешний заголовок Ethernet и пересылает кадр через таблицу маршрутизации. Выходной RB-коммутатор декапсулирует кадр TRILL и пересылает исходный кадр Ethernet. Кадры подразделяются на известные одноадресные и многоадресные кадры (неизвестные одноадресные, широковещательные и многоадресные кадры) [8].

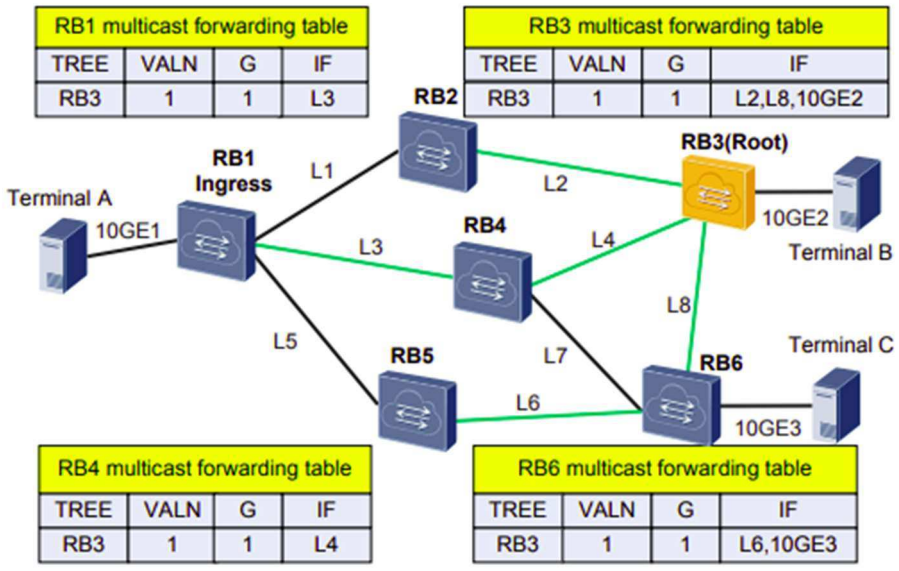

Рисунок 1.2.3 — Пример многоадресной рассылки кадра от хоста A

Входной RB пересылает неизвестный одноадресный кадр через таблицу маршрутов многоадресной рассылки (рисунок 1.2.3). После приема кадра выходной RB использует исходный MAC-адрес кадра и входной псевдоним для создания записи MAC-адреса для входного RB.

Здесь стоит обратить внимание на алгоритм пересылки многоадресного кадра. После приема такого кадра (неизвестный одноадресный, широковещательный или многоадресный кадр) входной RB выбирает корень дерева многоадресной передачи в соответствии с настроенными правилами, такими как распределение нагрузки на основе VLAN. Затем входной RB добавляет псевдоним корня в поле Egress RB заголовка TRILL, добавляет свой собственный псевдоним в поле Ingress RB и устанавливает бит M равным 1. Во внешнем заголовке Ethernet MAC-адрес назначения является фиксированным. на адрес All-RBridges 01-80-C2-00-00-40, и MAC-адрес источника изменяется на каждом транзитном RB.

Соответствующая многоадресная запись находится с использованием RB + VLAN + MAC, RB + VLAN и RB в качестве ключа по очереди. Кадр пересылается через порты, указанные в многоадресной записи. Если соответствующая многоадресная запись имеет флаг выхода, RB также декапсулирует кадр и направляет его в локальную сеть. На рисунке 1.2.3 показано, как кадр многоадресной рассылки пересылается через сеть TRILL.

Если это был неизвестный одноадресный кадр, то после обработки запроса-ответа входной RB и выходной RB изучают привязку MAC-RB и пересылают последующие кадры по одноадресному маршруту. Входной RB добавляет псевдоним RB, который соответствует MAC-адресу назначения, в поле Egress RB, добавляет свой собственный псевдоним в поле Ingress RB, устанавливает бит M в 0 и пересылает кадр.

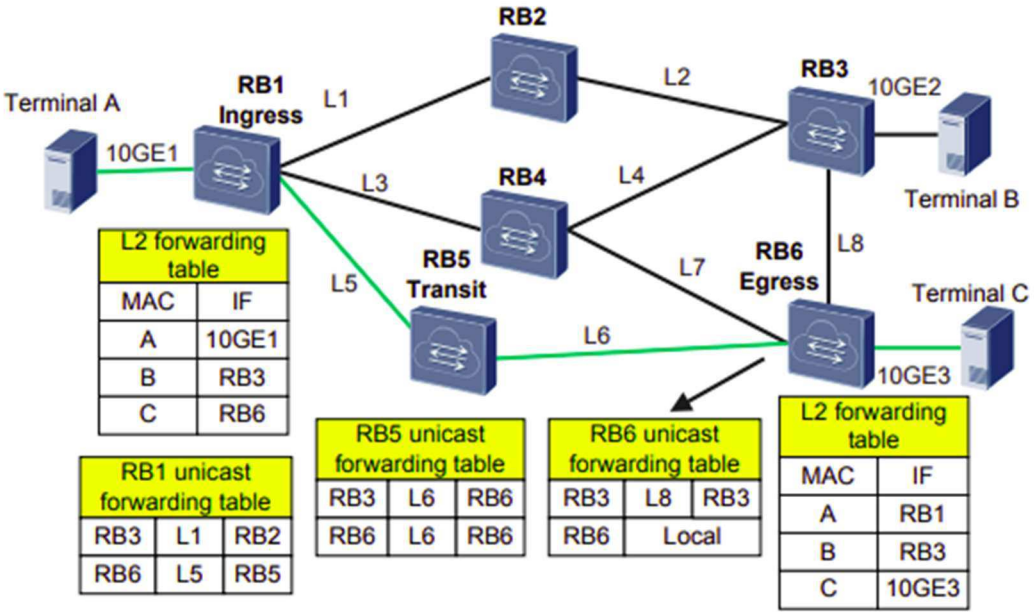

Рисунок 1.2.4 — Пример одноадресной пересылки и одноадресной таблицы маршрутизации

После приема известного одноадресного кадра транзитный RB проверяет, является ли он выходным RB в кадре (рисунок 1.2.4). Если да, то он декапсулирует кадр. Если нет, он использует выходной RB для поиска таблицы маршрутов одноадресной рассылки TRILL и таблицы следующего перехода, обновляет внешний заголовок Ethernet (изменяет MAC-адрес назначения на MAC-адрес следующего перехода и меняет исходный MAC-адрес на собственный MAC-адрес), и отправляет кадр через указанный порт. Во время пересылки в заголовке TRILL изменилось только поле Hop Count.

TRILL сочетает в себе преимущества коммутации второго уровня и маршрутизации третьего уровня, обеспечивая простое, масштабируемое и высокопроизводительное решение для центра обработки данных, где и используется в большинстве случаев. Главным достоинством можно считать устранение ограничений обычных сетей второго уровня на основе STP. Из недостатков можно отметить цену оборудования, использование инкапсуляции и существование служебного трафика.

1.3 Cisco FabricPath

Облачные технологии и STP (Spanning Tree Protocol) не могут удовлетворить требования современных центров обработки данных. Компания Cisco решила усовершенствовать TRILL (Transparent Interconnection of Lots of Links) тем самым разработав технологию FabricPath.

Cisco FabricPath — это технология, разработанная для преодоления ограничений STP, включая масштабируемость, конвергенцию и ненужное отключение соединений [9]. Она использует в основе своего алгоритма протокол и по той же причине эта технология называется маршрутизацией второго уровня.

Фактически, FabricPath является решением, позволяющим избежать использования STP, и поэтому рассматривается как замена протокола Spanning Tree.

В основе маршрутизации FabricPath лежит протокол маршрутизации 3 уровня IS-IS, который помогает строить топологии без петель. Этот протокол называется IS-IS уровня 2, который является модифицированной версией протокола IS-IS уровня 3. Таким образом, используются возможности протокола 3 уровня, но в самой сети может работать другой протокол (OSPF, RIP, BGP).

FabricPath позволяет создавать более гибкую структуру Ethernet, которая устраняет многие ограничения STP. Для поиска и выбора наилучшего пути или путей к заданному месту назначения в домене FabricPath использует протокол Shortest-Path First (SPF), входящий в состав используемого протокола маршрутизации IS-IS. Кроме того, технология FabricPath обеспечивает стабильность сети, и предоставляет масштабируемые аппаратные возможности обучения и пересылки, не связанные с программным обеспечением или производительностью процессора.

Преимущества развертывания сети на основе FabricPath включают в себя [9]:

- упрощение и снижение затрат на развертывание и эксплуатацию – FabricPath обеспечивает принцип «plug-and-play» с минимальной начальной конфигурацией. Развертывание сети становится проще, а устранение неполадок остается простым и прозрачным.
- максимальная гибкость – FabricPath устраняет многие конструктивные ограничения, связанные с STP, обеспечивая упрощенную мобильность рабочей нагрузки, приложения для кластеризации сети и расширение VLAN;
- увеличение доступной полосы пропускания - благодаря таким возможностям, как Equal-cost multipathing (ECMP) и multitopology-frowarding, FabricPath позволяет вам использовать всю доступную пропускную способность в сети;
- повышение доступности - FabricPath обеспечивает быструю повторную конвергенцию и изоляцию в области отказов, изолируя конечных пользователей и приложения от изменений в сети.

Другими словами, в то время как FabricPath приносит пользу не только серверам и приложениям, предоставляя прозрачную сетевую структуру, которая позволяет контролировать рабочую нагрузку и обеспечивает максимальную гибкость развертывания, но и значительно упрощая развертывание, конфигурацию и обслуживание сети для оперативного устранение неполадок.

Структуру FabricPath технологии можно представить в виде уровней, которые выполняют разные функции. Пример топологии с использованием данной технологии представлен на рисунке 1.3.1.

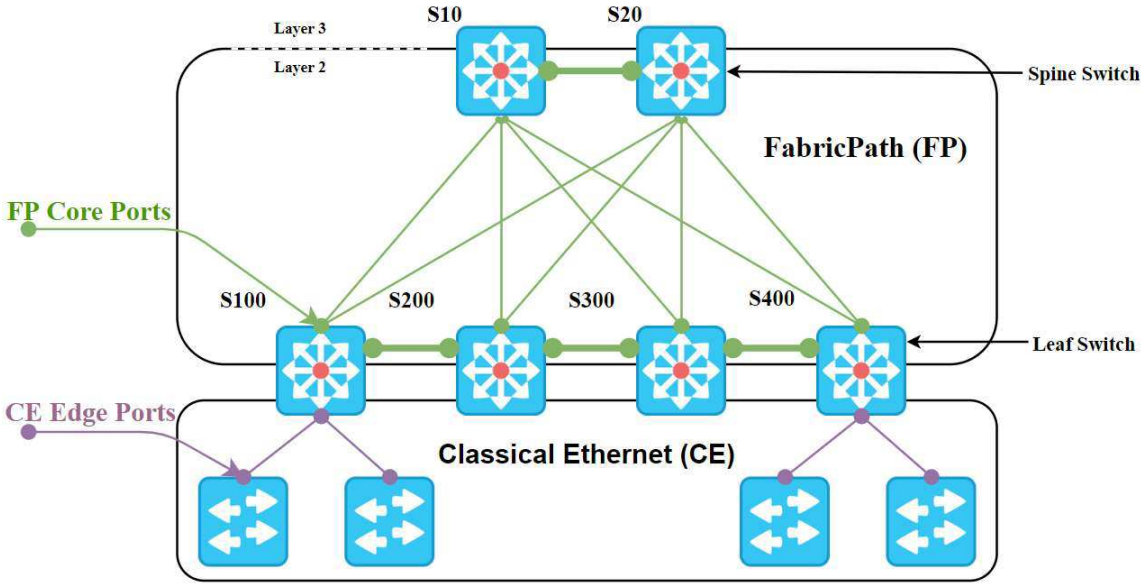

Рисунок 1.3.1 – Пример топологии с использованием технологии FabricPath

На рисунке 1.3.1 также представлены следующие термины:

- Spine Switch (S10, S20) – коммутаторы, предназначенные для транспортировки трафика внутри FP облака;
- FabricPath (FP) Core Ports – интерфейсы, подключенные к другому устройству в FP облаке. Основные функции: отправка и получение трафика с FP заголовками, обмен информацией о топологии с помощью L2 IS-IS, использование Switch ID Table (таблица 1.3.1) для маршрутизации трафика;
- Classical Ethernet (CE) Edge Ports – интерфейсы подключенные к обычным Ethernet устройствам. Основные функции: отправка и получение стандартных 802.3 Ethernet кадров, участие в STP, передача трафика при помощи таблицы MAC адресов;
- Leaf Switch (S100-S400) – приграничные коммутаторы, которые принимают обычный Ethernet трафик и навешивают FP заголовки.

Таблица 1.3.1 – Switch ID Table для коммутатора S100

Switch	Interface
S10	L1
S20	L2
S200	L1, L2, L3
…	…
S400	L1, L2, L3

Маршрут строится при помощи FabricPath Switch ID Table (таблица 1.3.1) и обычной таблицы MAC-адресов, которые хранятся на приграничных и Classical Ethernet устройствах [10]. Особенность в построении маршрута зависит от «стоимости» (метрики), которая хранится в Switch ID Table и может изменяться во время конфигурации в зависимости от требований. Стоимость соединения является статичной величиной, что означает ее независимость от задержек, пропускной способности канала и т.п. В любом случае, протокол IS-IS L2 будет строить наикратчайший маршрут, который напрямую зависит от метрик в сети. Если метрика маршрутов схожа, то используется балансировка трафика.

Перед передачей трафика из Classical Ethernet по FabricPath облаку приграничный (Leaf) коммутатор навешивает свои заголовки (рисунок 1.3.2) [10].

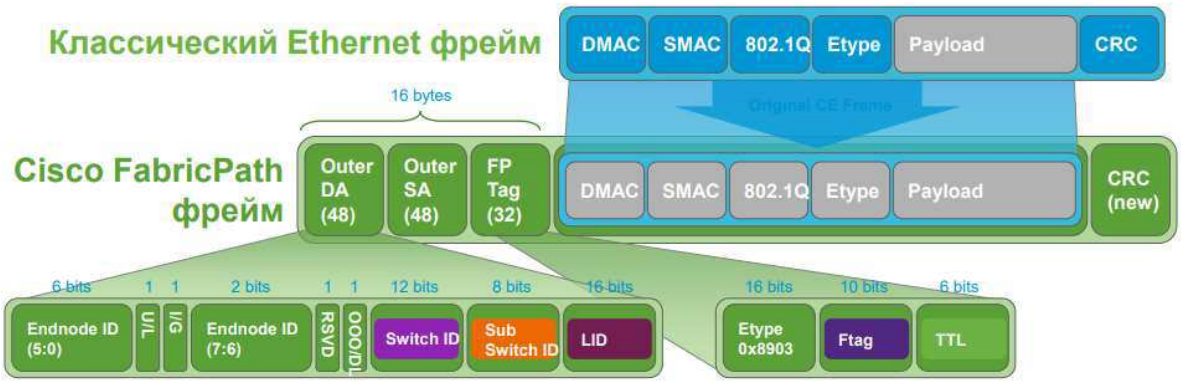

Рисунок 1.3.2 – Кадр FabricPath

FabricPath использует формат инкапсуляции MAC-in-MAC. К исходному кадру Ethernet, включая заголовок 802.1Q, добавляются 48-битный внешний адрес источника (OSA), 48-битный внешний адрес назначения (ODA) и 32-битный заголовк FabricPath. Хотя внешние SA и DA могут отображаться в виде 48-битных MAC-адресов, коммутаторы FabricPath, получающие такие кадры через основной порт FabricPath, анализируют эти поля в соответствии с форматом, показанным на рисунке 1.3.2.

Алгоритм коммутации пакетов данных внутри FP облака выглядит следующим образом:

- каждый коммутатор входящий в FP облако строит Switch ID (L2 маршрутизация) Table;
- входной (Leaf) коммутатор: по MAC-таблице определяет Switch-ID (SID), sub-Switch-ID (sSID) (для VPC технологии) и локальный ID или LID (он же выходной интерфейс). Для данного SID/sSID/LID определяется выходной интерфейс. Входной Ethernet пакет инкапсулируется;
- транзитный (Spine) коммутатор: для данного SID/sSID/LID определяется выходной интерфейс;
- приграничный (Leaf) коммутатор. Для данного SID/sSID/LID определяется выходной порт, пакет декапсулируется и передается как 802.3 Ethernet фрейм.

Полной таблицей маршрутизации владеют, в большинстве случаев, именно приграничные (Leaf) коммутаторы, а поскольку в реальных сетях количество MAC-адресов может исчисляться сотнями тысяч, то это накладывает свои ограничения по размеру таких таблиц [11].

Для борьбы с петлями, которые могут появляться, используется TTL, аналог TTL IP. Значение хопов по умолчанию 32. Т.е кадр может посетить FabricPath устройство лишь 32 раза.

Для работы с широковещательным и мультикаст трафиком используется multidestonation tree, которое затрагивает все коммутаторы в FP-облаке (рисунок 1.3.3).

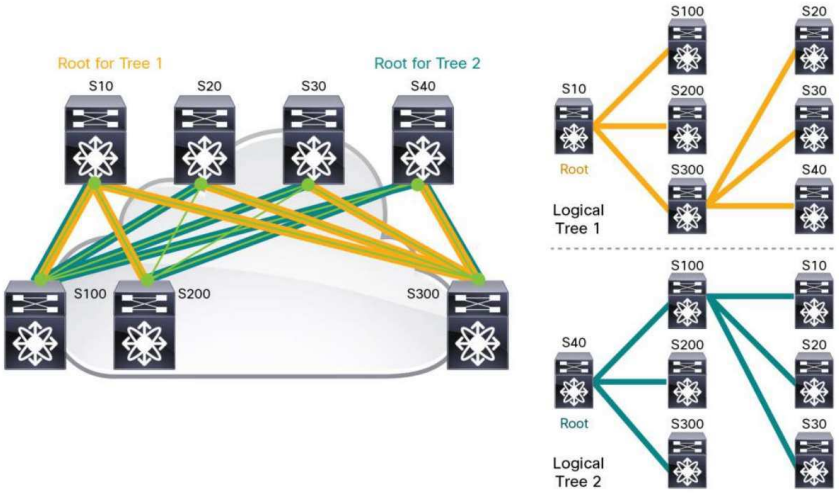

Рисунок 1.3.3 – Пример Multidestonation tree

Алгоритм работы FP с Broadcast и Multicast:

- для FP топологии строится 2 логических Multidestonation дерева
- корневой коммутатор выбирается для каждого multidestonation tree в FP домене
- беспетельное дерево строится для каждого корневого коммутатора, иначе говоря Ftag

Все оставшиеся соединения использоваться не будут.

Ftag это идентификатор дерева. От него зависит какой трафик будет в нем передаваться. Для Ftag1 это broadcast, multicast и fluding. Ftag2 используется для передачи multicast трафика. Это необходимо чтобы распределить нагрузку по физическим каналам внутри FP облака, т.е трафик будет хэшироваться в то или иное дерево. Выбор корневого коммутатора осуществляется по приоритету, если же он одинаков, то по SysID. Но все же лучшей практикой считается контролирование размещения корневого коммутатора. В случае выхода из строя

корневого коммутатора, начинается перерасчет деревьев. Среднее время сходимости 500 мс.

Преимущества FP:

- FP не использует STP, тем самым нивелируя его недостатки;
- простая конфигурация. Определяется кол-во VLAN в FP облаке и включаются FP на Core портах);
- практически абсолютная толерантность к топологии;
- реализация считается самой отказоустойчивой. На базе протокола STP такую топологию построить невозможно.

Недостатки:

- инкапсуляция, которая увеличивает нагрузку на оборудование;
- ограниченный набор устройств (L3 Cisco Nexus 5500, 6000, 7000);
- для некоторых устройств необходимы дополнительные модули;
- итоговая стоимость оборудования;
- проприетарность технологии;
- для борьбы с зацикливанием трафика используется TTL, который уменьшается при прохождении через устройство в FP облаке.

1.4 Балансировка трафика

Для многоканальной и отказоустойчивой передачи данных в сетях используется механизм ECMP (Equal-cost multi-path routing,). Главной особенностью ECMP является балансировка трафика по равноценным маршрутам. Данная функция позволяет распределить нагрузку равномерно по все доступным каналам связи в сети. Балансировка может осуществляться двумя способами:

- по пакетам;
- по потокам.

Балансировка по пакетам работает по принципу «один канал – один пакет». Данный подход является неоптимальным, потому что может возникнуть потеря пакетов, что негативно скажется на работе сети.

Балансировка по каналам использует алгоритм «один канал – один поток». Под потоком понимается совокупность параметров, которые могут однозначно определить пакеты, генерируемые теми или иными приложениями, протоколами, сервисами и т.п. Выбор канала для потока осуществляется при помощи хэш-функции по следующей формуле:

$$X = (\mathrm{SIP} + \mathrm{DIP} + \mathrm{Proto} + \mathrm{SPort} + \mathrm{DPort}) \,\%\, N;$$

где X – номер полученного канала;

SIP – IP адрес источника;

DIP – IP адрес назначения;

Proto – идентификатор протокола;

SPort – TCP\UDP порт источника;

DPort – TCP\UDP порт назначения;

N – количество доступных каналов.

Алгоритм балансировки по каналам работает следующим образом:

– Перед отправкой пакета алгоритм считает хэш-функцию из полей заголовков пакета;

– Полученная хэш-сумма делится на количество доступных путей до пункта назначения;

– По полученному числу определяется канал и в него отправляется пакет.

1.5 Выводы по главе 1

За последние десять лет были разработаны как минимум три протокола, которые позволяли бы использовать на втором уровне сети избыточные связи, тем самым нивелируя недостатки связанные с протоколами семейства STP и проблемами с циклами на втором уровне. Однако, все эти протоколы имеют одинаковые недостатки, поскольку работают на основе протокола состояния канала IS-IS. Это накладывает определенные ограничения на оборудование и на пропускную способность каналов. Так, например, инкапсуляция, которая используется в каждом вышеописанном протоколе, увеличивает нагрузку на центральные процессоры коммутаторов, что в свою очередь приводит к увеличению мощности последних и пропорциональному повышению стоимости оборудования. Еще одной важной проблемой инкапсуляции является ограничение по максимальному размеру полезного блока данных (Maximum transmission unit, MTU). Поскольку к исходному Ethernet кадру добавляются заголовки, то их размер компенсируется из исходного MTU, что приводит к его изменению для клиенткой и транспортной сети, а также к сопутствующим ограничениям [12]. Также стоит отметить, что наличие служебного трафика может привести к сбоям сети в случае их утери или искажении.

По результатам анализа вышеперечисленных протоколов было принято решение разработать аналог, который устранял бы их недостатки.

2 Проектирование алгоритма

2.1 Технические решения, использованные при разработке алгоритма

Алгоритм проектируемого протокола должен исключать недостатки вышеупомянутых решений. Для этого должен выполняться ряд следующих условий:

1. протокол не должен использовать служебный трафик;
2. протокол не должен использовать инкапсуляцию;
3. протокол должен бороться с возможностью появления BC-штормов и их последствий, т.е контролировать BC-трафик, не только путем использования TTL на пакете, как в случае ранее описанных решений.

Для разработки алгоритма, который выполнял бы вышеуказанные условия, был введен ряд терминов, описывающий некоторые новые функции и состояния коммутаторов и его интерфейсов:

- Edge интерфейс – интерфейсы, к которым подключаются конечные устройства, не входящие в работу протокола;
- Core интерфейс – интерфейсы, которые подключаются к другим коммутаторам из сети протокола;
- таблица MAC-адресов – отличается от классической таблицы MAC-адресов наличием поля Metric и возможностью хранить несколько записей для одного адреса, в случае если возможна балансировка;
- Metric – числовое значение в таблице MAC-адресов, влияющее на выбор маршрута для отправляемого трафика;
- заголовок кадра – набор байтов, содержащих какую-либо информацию. В данном случае, это информация о стоимости маршрута до устройства, с которого пришел кадр.

Для вставки заголовка, который содержит в себе метрику, были выделены 4 байта между адресом источника (SA) и исходным EtherType. Первые 2 байта занимает EtherType, который позволяет отличать Ethernet кадры проектируемого протокола от всех остальных. Для проверки работоспособности алгоритма и моделирования был выбран EtherType 8999, т. к. он является свободным [13]. Вторые 2 байта содержат в себе непосредственно метрику, которая влияет на то, как будут построены маршруты. Тем самым получается сохранить и использовать все исходные EtherType и заголовки, которые были вставлены ранее, например заголовок 802.1Q VLAN. Итоговый вид кадра представлен на рисунке 2.1.

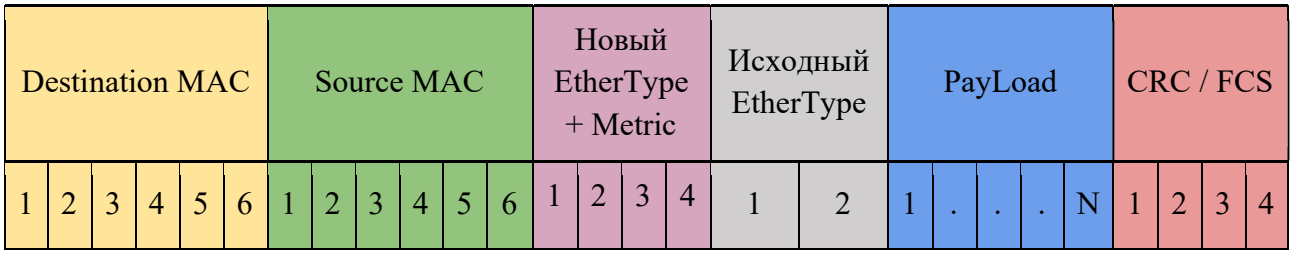

Рисунок 2.1 – Формат Ethernet кадра для работы протокола

По мере своей работы протокол должен анализировать кадры и хранить кратчайшие маршруты до существующих в сети узлов. Для этого создана специальная таблица, которая похожа на обычную таблицу MAC-адресов, но имеет дополнительное поле Metrics и возможность хранить для одного адреса несколько интерфейсов. Хранение нескольких интерфейсов обусловлено возможностью балансировать трафик, если позволяет топология. Пример таблицы представлен на рисунке 2.2.

MAC	Interface	Age	Metrics
A	e0/0	30	10
A	e0/1	30	10
B	e0/0	30	20
...	...	...	....

Рисунок 2.2 – Пример таблицы проектируемого протокола

Как видно из примера, для устройства с MAC-адресом A возможна балансировка трафика, т. к. все условия соблюдены (интерфейсы разные и метрика одинакова).

С полным алгоритмом работы проектируемого протокола можно ознакомиться на рисунке 2.3.

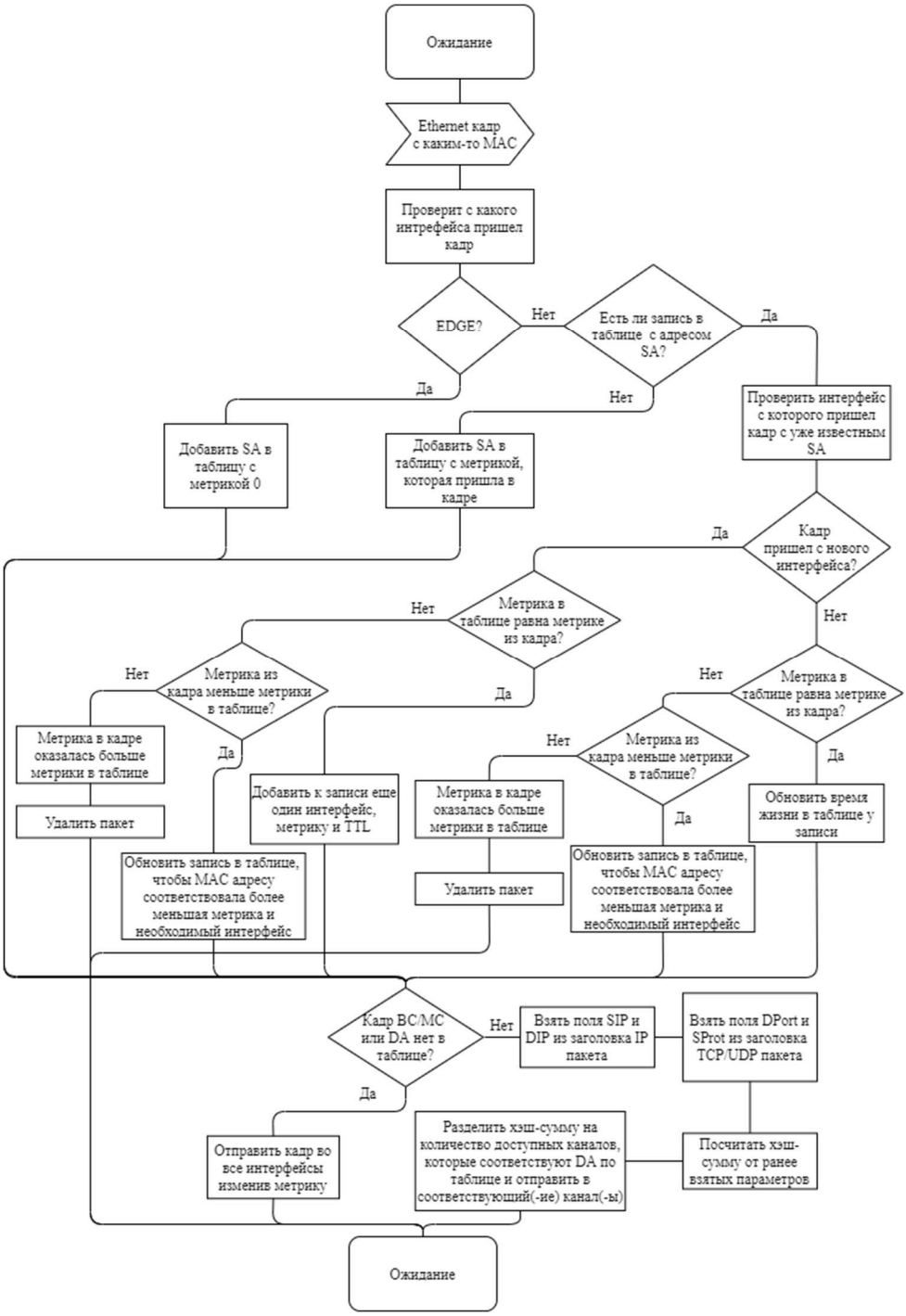

Рисунок 2.3 – Алгоритм проектируемого протокола

Особое внимание стоит уделить изучению пакетов. Данный алгоритм подразумевает использование маршрутов с минимальными метриками. Поэтому пакеты пришедшие с минимальным значением моментально меняют запись в MAC таблице на наилучшую. В случае, если приходит пакет с большим значением, то он удаляется. Это также помогает решать проблемы колец и BC-трафика, поскольку приходящий с большой метрикой трафик считается излишним и уничтожается. Если же из-за неполадок в сети канал с минимальной метрикой становится недоступен, то после истечения срока жизни (по умолчанию 30 секунд) записи в таблице, которые находились за недоступным интерфейсом, удалятся и трафик будет направлен во все порты, пока топология не восстановится.

В случае, если метрики с нескольких интерфейсов одинаковы протокол может использовать балансировку.

Все этапы и получаемые маршруты можно рассмотреть на примере небольшой сети (рисунок 2.4).

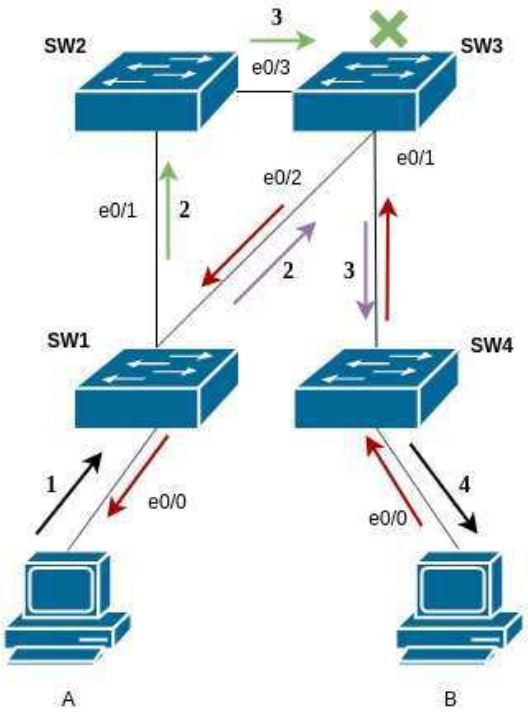

Рисунок 2.4 – Небольшая сеть для примера

Для примера предположим, что с устройства A был запущен Ping в сторону устройства B. Поскольку это Ethernet сеть, то будет сформирован ARP-запрос. Отправившийся широковещательный Ethernet кадр (черная стрелка 1) к коммутатору SW1 будет изучен и SA будет добавлен в MAC таблицу с метрикой 0. После этого кадру будет присвоен новый EtherType. Поскольку кадр является широковещательным, он будет разослан на все интерфейсы коммутатора SW1 (кроме того, откуда пришел трафик) при этом увеличив метрику на заданное число (зеленая и фиолетовая стрелки 2). Далее трафик направляется на соседние коммутаторы попутно увеличивая метрику (зеленая и фиолетовая стрелки 3). Трафик прошедший по маршруту A-SW1-SW2-SW3 после анализа будет уничтожен (зеленый крестик), поскольку в таблице MAC-адресов существует более короткий путь A-SW1-SW3. Дойдя до коммутатора SW4 трафик также будет разослан на все интерфейсы, но поскольку к одному из них подключено конечное устройство, то порт работает в режиме Edge, а значит перед отправкой необходимо снять присвоенный EtherType и отправить кадр устройству B (черная стрелка 4). Обратный же путь будет проделывать уже Unicast кадр, а т.к предыдущий кадр проходя через коммутаторы анализировался и записывался в таблицу, то и ответ на ARP-запрос пойдет по кратчайшему пути (красные стрелки).

2.2 Выводы по главе 2

На основе результатов анализа предметной области и решений, которые предлагают вендоры сетевых технологий, был разработан алгоритм протокола для использования всех доступных каналов на втором уровне. Разработанный протокол отличается отсутствием инкапсуляции и служебного трафика. Помимо этого, он выполняет функции, которые позволяют использовать все доступные канала между коммутатора и бороться с последствиями колец и BC-трафика. Это

достигается благодаря анализу Ethernet кадров и сравнению метрики внутри с метрикой в таблице MAC-адресов на коммутаторе.

Следующим этапом в разработке протокола будет его реализация и тестирование на специально подготовленном стенде, которое должно подтвердить все заявленные функции.

3 Реализация и тестирование протокола

3.1 Реализация протокола

3.1.1 Python

Python – это универсальный современный ЯП высокого уровня, к преимуществам которого относят высокую производительность программных решений и структурированный, хорошо читаемый код. Синтаксис Питона максимально облегчен, что позволяет выучить его за сравнительно короткое время. Ядро имеет очень удобную структуру, а широкий перечень встроенных библиотек позволяет применять внушительный набор полезных функций и возможностей. ЯП может использоваться для написания скриптов, прикладных приложений, а также разработки WEB-сервисов.

Python может поддерживать широкий перечень стилей разработки приложений, в том числе, очень удобен для работы с ООП и функционального программирования.

Один из самых популярных интерпретаторов языка – CPython, написанный на Си. Распространяется эта среда разработки бесплатно по свободной лицензии. Интерпретатор поддерживает большинство популярных платформ.

Именно благодаря простоте написания и отладки кода для реализации протокола был выбрал язык программирования Python.

3.1.2 Структура кода протокола

Код реализованного протокола можно разделить на две части:

– блок изучения кадра;

– блок управления пересылкой кадров.

Блок изучения кадра в свою очередь можно разделить на следующие составные:

– блок изучения метрик и заполнения таблиц;

– блок балансировки трафика.

Основной задачей блока изучения кадров является заполнение таблиц маршрутизации. Каждый кадр, который проходит через коммутатор, изучается и информация помещается в специальную таблицу, содержащую MAC-адрес источника, интерфейс, с которого пришел кадр, «возраст» записи (по умолчанию 30 секунд) и метрику. Записи с одинаковым MAC-адресом источника попадают в таблицу с заменой, которая отрабатывает по алгоритму «меньше метрика – лучше маршрут». Если у таких записей метрика одинакова, но отличаются входные интерфейсы, то в таблицу добавляются обе записи. Это позволяет в дальнейшем использовать балансировку трафика.

Блок балансировки трафика необходим для разделения трафика по доступным каналам передачи данных. Для определения порядкового номера интерфейса, в который можно отравить кадр, используется формула, представленная в главе 1.4. Хэш-сумма, полученная путем суммирования полей заголовков кадра, делится на количество доступных интерфейсов, за которыми находится устройство с MAC-адресом назначения. Тем самым получается добиться распределения потока трафика на все доступные каналы.

После изучения кадра и получения номера интерфейса срабатывает блок пересылки кадра. Если кадр был получен от устройств, которые не входят в сеть коммутаторов, где работает реализуемый протокол, то на интерфейсе, откуда пришел данный кадр в таблице ставится метрика 0, а в сам кадр вставляется EtherType 8999. Данный EtherType необходим для упрощения поиска метрики в кадре и для возможности отличить пакеты протокола от других пакетов в сети. Если пакет пришел с другого коммутатора сети протокола, то происходит проверка записи в таблице на возможность ее изменения и увеличив метрику на заданную в настройках единицу. При выходе пакета из сети работы протокола EtherType и метрика с кадра снимаются.

3.2 Тестирование протокола

3.2.1 Описание стенда для тестирования

Для тестирования функционала разработанного протокола был создан виртуальный стенд, который состоит из тринадцати виртуальных машин на базе ОС Linux. Семь машин работают в режиме коммутатора, т.е. на них запускается разработанный протокол. Остальные машины работают в простом режиме. Они необходимы для запуска программ, которые могут проводить диагностику сети (ping, netcat).

Все устройства соединяются при помощи виртуальных каналов передачи данных. Итоговый вид топологии изображен на рисунке 3.1.

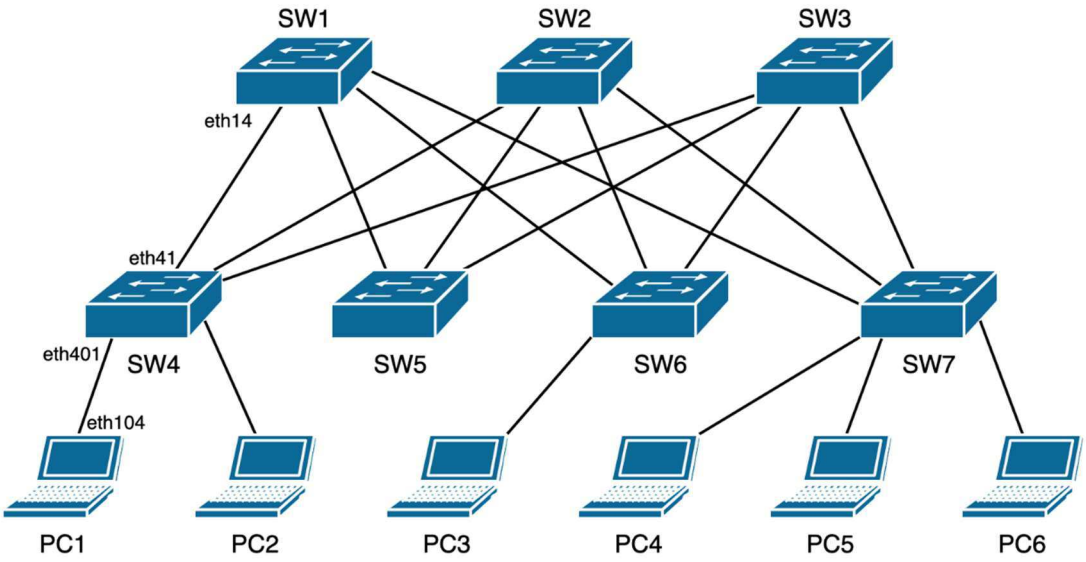

Рисунок 3.1 – Итоговый вариант стенда для тестирования

Данный виртуальны стенд упрощает запуск и отладку разработанного протокола в сравнении со стендом из физический коммутаторов и компьютеров.

3.2.2 Результаты тестирования разработанного протокола

Для тестирования протокола на выполнение заявленных при разработке функций на стенде использовались команды ping и netcat. Команда ping генерирует ICMP пакеты, которые позволяют узнать о доступности того или иного хоста в сети. Команда netcat позволяет создавать потоки данных на основе протоколов UDP и TCP. Сеть построена с использованием IPv6, что облегчает тестированием при некоторых сценариях.

На данном стенде было проведено тестирование разработанного протокола, которое показало следующие результаты. На рисунке 3.2 изображена топология с работающими на время тестирования устройствами.

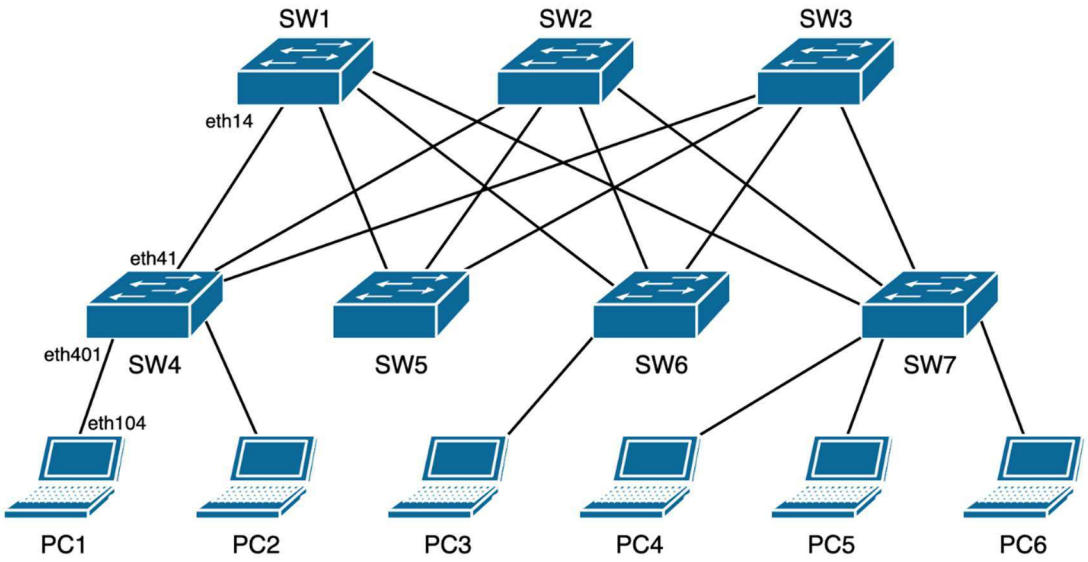

Рисунок 3.2 – Стенд с полученными результатами

При запуске ping с PC2 на IPv6 multicast-группу ff02:1 были получены следующие результаты:

– ответ на ping был получен от всех конечных хостов;

– задвоений и потерь пакетов не наблюдался;

– хосты-коммутаторы отработали алгоритм протокола без ошибок и корректно заполнили таблицы маршрутизации.

На рисунках 3.3 и 3.4 представлена информация с PC2 и коммутаторов соответственно.

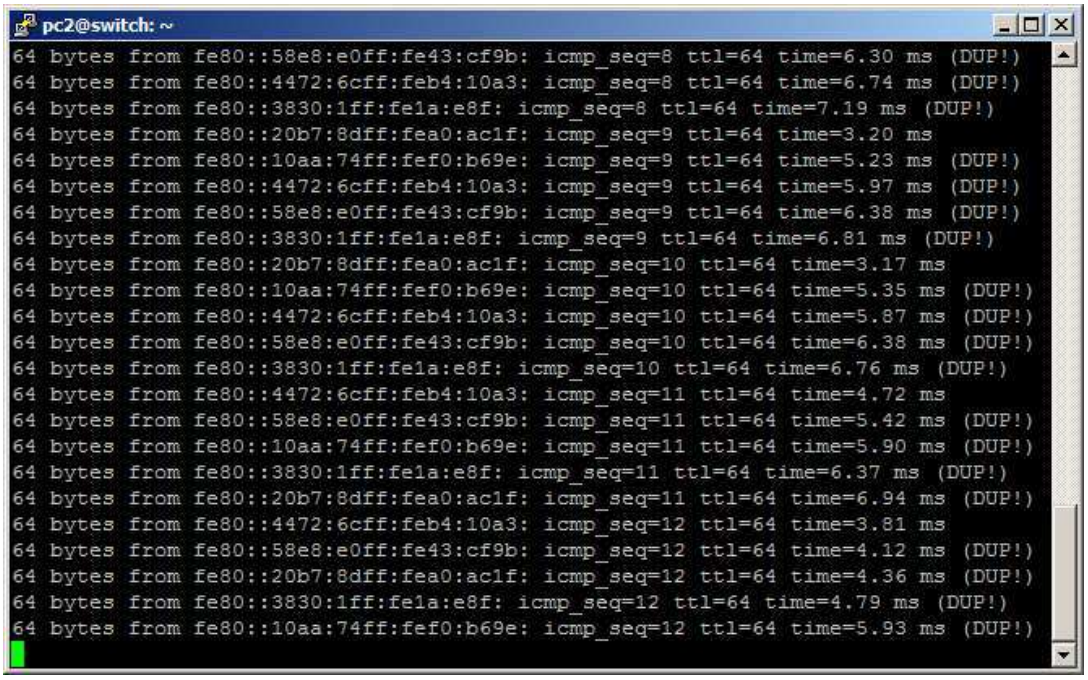

Рисунок 3.3 – Результаты выполнения команды ping на PC2

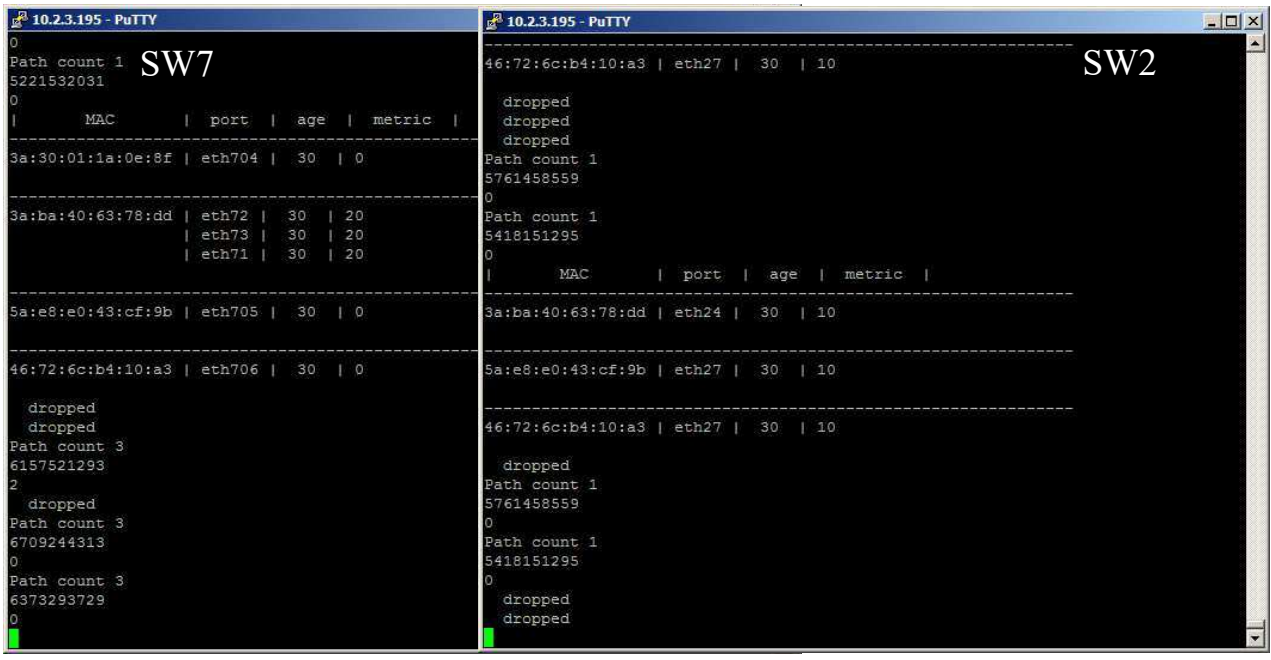

Рисунок 3.4 – Информация с коммутаторов SW7 и SW2 при выполнении команды ping на PC2

На примере результатов с рисунка 3.4 можно проследить полученные при работе протокола маршруты:

– для хостов PC5 и PC6: PC2 ⇔ SW4 ⇔ SW2 ⇔ SW7 ⇔ PC5/PC6;

– для хоста PC4: PC2 ⇔ SW4 ⇔ SW1 ⇔ SW7 ⇔ PC4.

Данные результаты позволяют говорить о том, что алгоритм протокола работает правильно и разделяет трафик по потокам.

Для дополнительной проверки был проведен эксперимент с командой netcat. Результаты работы команды и алгоритма представлены на рисунке 3.5.

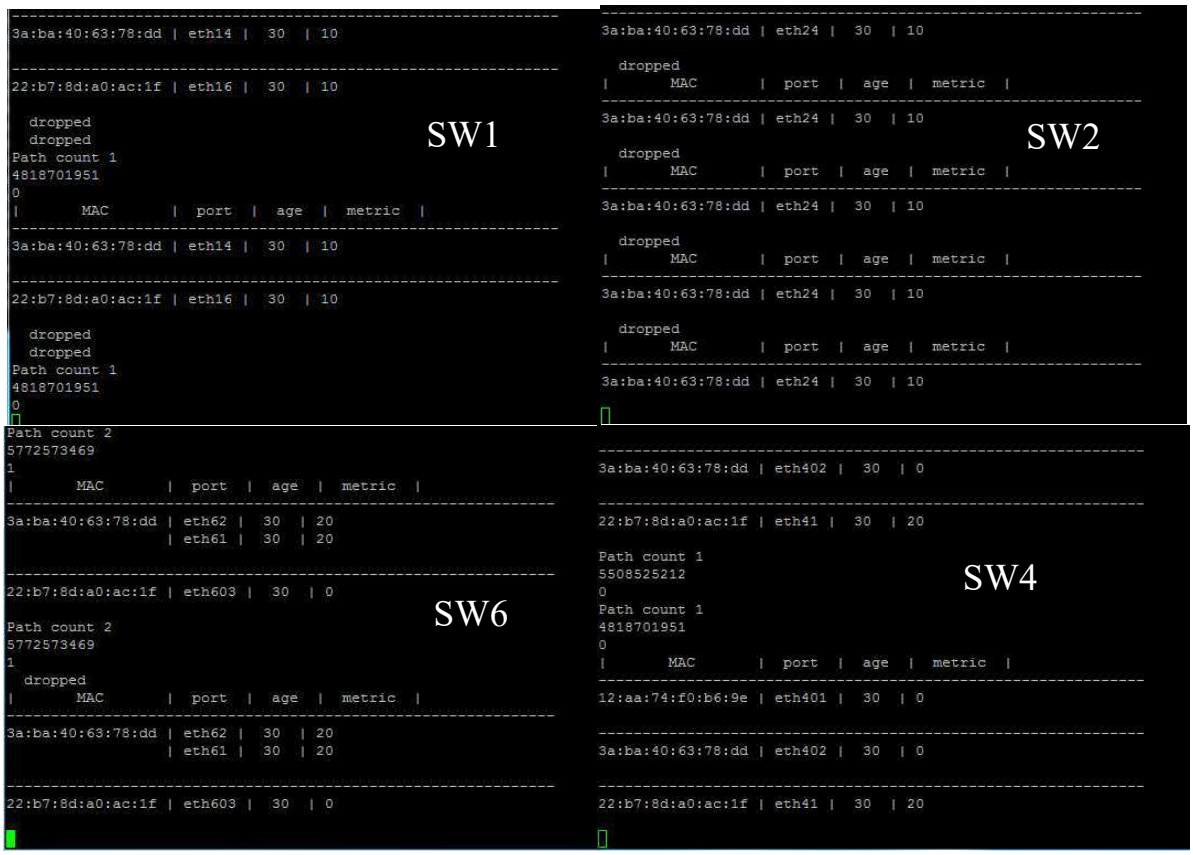

Рисунок 3.5 – Информация с коммутаторов при работе команды netcat

Рисунок 3.6 может в более наглядном виде увидеть трафик от PC2 до PC3. На рисунке видно, что пакеты от PC2 пошли по всем каналам, т.е широковещательно. Это свойственно ARP-запросу, который позволяет связать IP и MAC адреса машин в сети. В результате работы блоков предотвращения

петель и распределения трафика поток PC2⇒PC3 проходит по одним каналам сети, а поток PC3⇒PC2 по другим. При этом не создаются коллизии трафика и в таблицах коммутации всегда присутствуют альтернативные маршруты, которые могут быть использованы если сгенерировать больше уникальных по ключевым параметрам (SIP, DIP, Proto, SPort, DPort) пакетов.

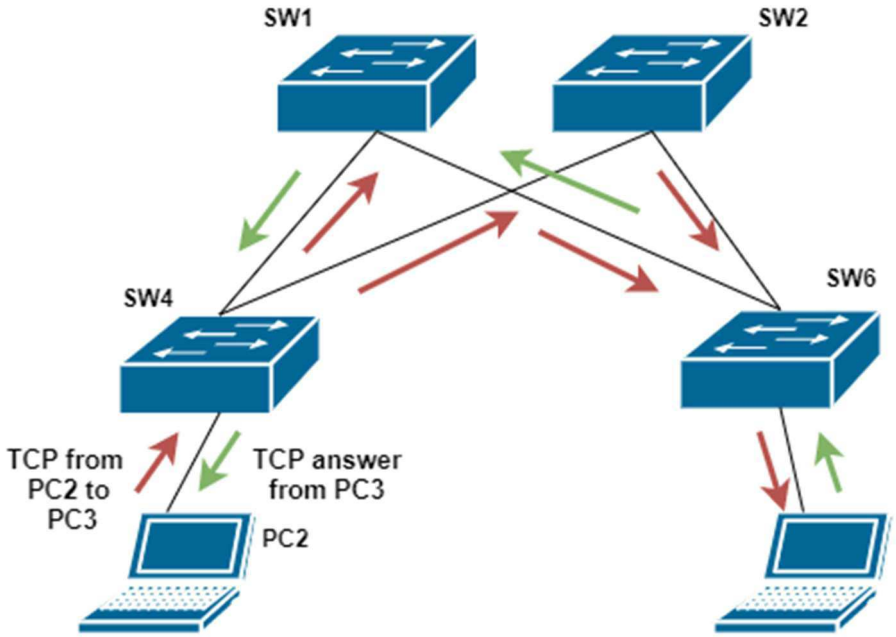

Рисунок 3.6 – Схема использования каналов при работе алгоритма и команды netcat

Из полученных результатов можно сделать вывод, что разработанный протокол выполняет заявленную функцию, а именно позволяет на уровне коммутации использовать все доступные каналы передачи данных, без риска выхода из строя сети.

ЗАКЛЮЧЕНИЕ

В процессе выполнения работы были исследованы технологии, которые позволяют строить полносвязанные сети на основе Ethernet. Пояснительная записка содержит информацию об этих технология, их достоинства и недостатки. Полученная информация была использована для разработки авторского алгоритма, который также позволяет строит подобные сети, но использует другой алгоритм построения маршрутов.

Реализованный алгоритм позволяет уменьшить размер заголовков с 40-60 байт до 4 байт, что благоприятно сказывает на размере полезной нагрузки Ethernet пакета. Помимо этого, разработанный алгоритм позволяет избавиться от служебного трафика. Вся необходимая информация, которая нужна для построения маршрутов, передается в Ethernet пакетах с полезной нагрузкой.

Для тестирования работы алгоритма, был написан код на языке программирования Python и запущен на виртуальных машинах Linux. Семь машин имитировали работу коммутаторов, а шесть конечных устройств. По итогам тестирования было выявлено, что разработанный алгоритм справляется с заявленными функциями и позволяет строить полносвязанные сети на базе коммутаторов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ковальков, Д. А. Обеспечение и методика расчета надежности узла предоставления мультисервисных телекоммуникационных услуг. / Д. А. Ковальков, Е. А. Гаврилин, О.С. Галин, Р. Д. Антонов, // Труды Международного симпозиума «Надежность и качество». − 2017. − №1. − С. 123-126.

2. Shortest Path Bridging (802.1aq) Technical Configuration Guide : техн. информация : NN48500-617, версия 2.3 / Extreme Networks, Inc. // Extreme Networks, Inc. [Электронный ресурс]. − Сан-Хосе, 2017. − Режим доступа: https://downloads.avaya.com/css/P8/documents/100128115

3. SPB Technology White Paper : техн. информация / New H3C Technologies Co. // H3C Technologies Co. [Электронный ресурс]. − Гонконг, 2018. − Режим доступа: http://www.h3c.com/en/Product_Technology/Operating_System/ComwareV7/Data_Center/White_Papers/201808/1102743_294549_0.html

4. IEEE 802.1aq [Электронный ресурс]. − Режим доступа: https://en.wikipedia.org/wiki/IEEE_802.1aq

5. TRILL (computing) [Электронный ресурс]. − Режим доступа: https://en.wikipedia.org/wiki/TRILL_(computing)

6. TRILL Technology White Paper : техн. информация / Huawei Technologies Co. // Huawei Technologies Co., Ltd [Электронный ресурс] − Шэньчжэнь, 2016. − Режим доступа: https://actfornet.com/ueditor/php/upload/file/20181230/1546123999973133.pdf

7. Багинян А., Долбилов А., Кашунин И., Кореньков В. Балансировка трафика в высоконагруженных системах с помощью протокола TRILL / А. Багинян, А. Долбилов, И. Кашунин // T-Comm: Телекоммуникации и транспорт. –2017. – №4. – С. 14-19.

8. TRILL Technology White Paper : техн. информация / New H3C Technologies Co. // H3C Technologies Co. [Электронный ресурс]. − Гонконг, 2018.

– Режим доступа: http://www.h3c.com/en/Product_Technology/Operating_System/ComwareV7/Data_Center/White_Papers/201808/1102744_294549_0.htm

9. Cisco FabricPath [Электронный ресурс]. − Режим доступа: https://www.cisco.com/c/dam/en/us/products/collateral/switches/nexus-7000-series-switches/at_a_glance_c45-605626.pdf

10. Cisco Nexus 5000 and 6000 Series NX-OS FabricPath Operations Guide : техн. информация : OL-28442-02, версия 6.0 / Cisco Systems, Inc. // Cisco Systems, Inc. [Электронный ресурс]. − Сан-Хосе, 2013. − Режим доступа: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus5000/sw/operations/farbric_path/602_n1_1/Cisco_Nexus_5000_6000_NX-OS_FabricPath_Operations_Guide.pdf

11. Cisco Nexus 7000 Series NX-OS FabricPath Configuration Guide : техн. информация : OL-22842-03 / Cisco Systems, Inc. // Cisco Systems, Inc. [Электронный ресурс]. − Сан-Хосе, 2019. − Режим доступа: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/sw/6_x/nx-os/fabricpath/configuration/guide/b-Cisco-Nexus-7000-Series-NX-OS-FP-Configuration-Guide-6x.html

12. Калашников, С. Современные оверлейные технологии для ЦОД. / С. Калашников // Хабр [Электронный ресурс] – Режим доступа: https://habr.com/ru/search/?q=%5Bоверлейные%20технологии%5D&target_type=posts

13. IEEE 802 Numbers [Электронный ресурс]. – Режим доступа: https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml

ПРИЛОЖЕНИЕ А

Листинг кода разработанного протокола

```
# Import libraries
from socket import *
import struct
import thread
import binascii
import os
import time
# Enumerate network interfaces
interfaces=os.popen("ip a | egrep -o '(eth|backup)[0-9]+'").read().split('\n')
interfaces.pop()
# Create sockets
sockets=[None]*len(interfaces)
for i in range(0,len(interfaces)):
  sockets[i] = socket(AF_PACKET, SOCK_RAW, htons(3))
  sockets[i].bind((interfaces[i],0))
print 'Switching on '+' '.join(interfaces)
# tag format
tag='\x89\x99\0\x0a'
# mac address table
mac_address_table = {}
def sum_bytes(frame):
  return int(frame.encode('hex'),16)
def print_mac(frame):
  return  binascii.hexlify(frame[0]) + ':' + binascii.hexlify(frame[1]) + ':' + binascii.hexlify(frame[2]) + ':' +
binascii.hexlify(frame[3]) + ':' + binascii.hexlify(frame[4]) + ':' + binascii.hexlify(frame[5])
def print_mac_address_table():
  for mac, data in mac_address_table.items():
    print '-----------------------------------------------------------'
    node = print_mac(mac)
    for inf in data:
```

```
    node = node + ' | '+str(inf[0].getsockname()[0])+' |  '+str(inf[1])+'  | '+str(inf[2])+ '\n'+'                '
  print (node)
########################################
# Determine where frame to be switched #
########################################
def dispatch_frame(frame,socket_in):
 sockets_out=list()
 if len(socket_in.getsockname()[0])==6: # edge port ?
  mac_address_table[frame[6:12]]=[[socket_in,30,0]]
 else:
  metric = int(frame[15:16].encode('hex'),16) # get metric from tag
  if mac_address_table.get(frame[6:12],'null')=='null':# Check MAC unknown or not
   mac_address_table[frame[6:12]]=[[socket_in,30,metric]]
  else:
   data=mac_address_table.get(frame[6:12])
   index = [i for i, (inf, _, _) in enumerate(data) if inf == socket_in]
   if index: # learn shorter route check socket equals
    if metric==data[index[0]][2]:
     data[index[0]]=[socket_in,30,metric]
     mac_address_table[frame[6:12]]=data
     #print_mac(frame[6:12])
     #print_mac(data[0])
     if int(frame[0].encode('hex'),16)%2 and socket_in!=data[0][0]:
      return list()
    elif metric>data[index[0]][2]:
     return list()
    else:
     mac_address_table[frame[6:12]]=[[socket_in,30,metric]]
   else: # sockets not equals
    if ([i for i, (_, _, cost) in enumerate(data) if cost==metric]):
     data.append([socket_in,30,metric])
     mac_address_table[frame[6:12]]=data
```

```
      elif ([i for i, (_, _, cost) in enumerate(data) if cost<metric]):
        return list()
      else:
        mac_address_table[frame[6:12]]=[[socket_in,30,metric]]
  if int(frame[0].encode('hex'),16)%2 or mac_address_table.get(frame[0:6],'null')=='null' :
    for socket in sockets:
      if socket!=socket_in:
        sockets_out.append(socket)
  else:
    path_count = len(mac_address_table.get(frame[0:6]))
    print 'Path count '+str(path_count)
    hash = sum_bytes(frame[30:34]) + sum_bytes(frame[34:38]) + sum_bytes(frame[27]) +
sum_bytes(frame[42:46]) + sum_bytes(frame[46:50])
    print (hash)
    path = hash%path_count
    print (path)
    sockets_out.append(mac_address_table.get(frame[0:6])[path][0])
  return sockets_out
########################################
# Main switch function
def switch(socket_in,interface):
  while 1:
    #receive frame
    frame = socket_in.recv(9014)
    #print 'Received frame on ' + socket_in.getsockname()[0] + ' from ' + binascii.hexlify(frame[6]) + ':' +
binascii.hexlify(frame[7]) + ':' + binascii.hexlify(frame[8]) + ':'+binascii.hexlify(frame[9]) + ':' +
binascii.hexlify(frame[10]) + ':' + binascii.hex$
    #determine where to send
    sockets_out=dispatch_frame(frame,socket_in)
    #send frame copies
    if len(sockets_out)==0:
      print '  dropped'
```

```
else:
 for socket_out in sockets_out:
   # if out socket NOT EDGE tag and to switch from switch add tag
   if len(socket_out.getsockname()[0])!=6:
     # if frame without tag
     if frame[12:14]!=tag[0:2]:
       if len(socket_out.getsockname()[0])==9:
         new_metric_bytes = struct.pack('h',50) # Add metric 50
         frame2=frame[0:12]+tag[0:2]+new_metric_bytes[1]+new_metric_bytes[0]+frame[12:len(frame)]
       else:
         new_metric_bytes = struct.pack('h',10) # Add metric 10
         frame2=frame[0:12]+tag[0:2]+new_metric_bytes[1]+new_metric_bytes[0]+frame[12:len(frame)]
     # if frame with tag
     else:
       # if out_port backupX0Y
       if len(socket_out.getsockname()[0])==9:
         new_metric = int(frame[15:16].encode('hex'),16)+50 # Add metric 50
         new_metric_bytes = struct.pack('h',new_metric)
         frame2=frame[0:14]+new_metric_bytes[1]+new_metric_bytes[0]+frame[16:len(frame)]
       # if out_port ethXY
       else:
         new_metric = int(frame[15:16].encode('hex'),16)+10 # Add metric 10
         new_metric_bytes = struct.pack('h',new_metric)
         frame2=frame[0:14]+new_metric_bytes[1]+new_metric_bytes[0]+frame[16:len(frame)]
   # if frame with tag for PC/Other
   else:
    if len(socket_out.getsockname()[0])==6 and frame[12:14]==tag[0:2]:
     #remove tag
     frame2=frame[0:12]+frame[16:len(frame)]
    else:
     #do not alter tags
     frame2=frame
```

Окончание приложения А

```
        #print ' sent to '+socket_out.getsockname()[0]
        socket_out.send(frame2)
# Start main switch function in separate threads
for i in range(0,len(interfaces)):
  thread.start_new_thread(switch,(sockets[i],''))
# Main thread will do nothing
while 1:
  time.sleep(1)
  #print_mac_address_table()
  # mac timeout
  print '|      MAC      |'+'  port  |'+'  age  |  metric  |'
  print_mac_address_table()
  for mac, data in mac_address_table.items():
    for inf in data:
      inf[1]=inf[1]-1
      if inf[1]==0:
        data.remove(inf)
    if len(data)==0:
      del mac_address_table[mac]
```

Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ

Заведующий кафедрой

_____ О.В Непомнящий
подпись инициалы, фамилия

«___» _____ 20___ г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Построение полносвязанных сетей на базе коммутаторов Ethernet
Тема

09.04.01 «Информатика и вычислительная техника»
код и наименование направления

09.04.01.05 «Сети ЭВМ и телекоммуникации»
код и наименование магистерской программы

Научный руководитель _____ доцент, канд. техн. наук Ф.А. Казаков
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ М.С. Русин
подпись, дата инициалы, фамилия

Рецензент _____ доцент, канд. техн. наук В.Н. Коршун
подпись, дата должность, ученая степень инициалы, фамилия

Нормоконтролер _____ доцент, канд. техн. наук Ф.А. Казаков
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2020